A Survey on Big Data Management and Job Scheduling

Sreedhar C.
CSE Department
G Pulla Reddy Engineering College
Kurnool. India

N. Kasiviswanath, PhD
Prof. & HOD, CSE Department
G Pulla Reddy Engineering College
Kurnool. India

P. Chenna Reddy, PhD
Professor
JNTU Pulivendula
Pulivendula. India

ABSTRACT
Big data has gained its popularity in the recent years due to the fact that there is a need for sophisticated method to collect, process, analyze and visualize huge volumes of data generated by our digital and computing world. Several challenges in handling petabytes of information, commonly named as Big data needs to be addressed in more efficient way. Big data management (BDM) is the process of collecting, storing, analysing and visualization of large volumes of data, which can be in the form of structured, unstructured and semi-structured formats. Problems such as data acquisition, data storage, data retrieval, data analysis, and data visualization can no longer be handled by traditional database systems. The primary purpose of this paper is to provide a comprehensive survey on Big data management and to provide an overview on various algorithms related to job scheduling in Hadoop and the latest advancements. These research directions can lead to exploration of Big data domain and result in development of optimal techniques and scheduling algorithms to address problems faced in Big data.

General Terms
Survey on Big data and Job Scheduling

Keywords
Big data, Big data management, Job Scheduling, Hadoop, MapReduce.

1. INTRODUCTION
The roots of big data have already spread into our planet. Data is being produced at an ever increasing rate. This growth in data production is being driven by organizations, individuals; switch from analogue to digital technologies, social media, Internet of Things and sensors. The rate at which the data produced in the recent years is far unexpected. Big data is a popular term used to describe the exponential growth and availability of data, both structured and unstructured. It is mainly collection of data sets so large and complex that is very difficult to handle them using traditional database management tools. Big data has become one of the important elements of business analytics, which provides tremendous opportunities for enterprise information management and decision making. In the recent study on the evolution of big data shows that big data is not only limited to business needs but also helps in research and scientific issues [29]. The rate at which the amount of data collected needs to address several issues and challenges such as real-time reports, transfer speed, unstructured data, scheduling of jobs and security issues [30]. Modern information technology is becoming the engine of operation and development of all walks of life. But the engine is facing a huge test of big data [39]. Big data management (BDM) is the process of collecting, storing, analysing and visualization of large volumes of data, which can be in the form of structured, unstructured and semi-structured formats. Table 1 describes about data diversity in the data sources.

Table 1. Data Sources and Formats

Data Source	Data Formats
Wikipedia data set	Un-structured
Amazon Movie Reviews	Semi-structured
Google Social data sets	Un-structured
Facebook Social data sets	Un-structured
E-commerce Transaction data sets	Structured
Job Resume	Semi-structured

The advances in information technology have created huge volumes of data. Social media is contributing a lot to this huge volume of data. Facebook produces more than 10 TB of data every day and Large Hadron Collider produces 15 PB of information every year. In order to handle such petabytes of information, Big data management tools has predominant role, which are capable of collecting the data, processing, reporting and visualizing in appreciable time. Relational database systems tools are developed to tackle the problems of data storage, retrieval, processing and querying with an assumption that data is structured. Various business data is breaking out in the form of geometric series [40] [5], problems such as collection, storage, retrieval, analysis, application and so on, can no longer be solved by the traditional information processing technology With the rapid development of technology, data is no longer in structured format. The data can be in the form of audio, video, web logs, images, system logs, network logs and in XML form. There is a need for newer means of data processing and storage to integrate these forms of data. Traditional relational database management systems in general use a centralized storage system and processing the data is performed without the use of distributed architecture. The tools used to handle huge volumes of data such as Hadoop uses distributed architecture. In the recent years, there is much advancements in data acquisition, data storage and processing technologies and this has led to the data generated by organizations is modified [31].

This paper presents: (a) comprehensive survey on Big data management; (b) related technologies; (c) challenging issues in Hadoop; (d) detailed study on current job scheduling

algorithms; (e) latest improvements made on scheduling algorithms.

The rest of the paper is organized as follows. Section 2 describes the concepts of Big data management; Section 3 presents technologies involved in Big data; Section 4 explains the issues and challenges faced by growth of the data; Section 5 introduces various algorithms adopted related to job scheduling in Hadoop; Section 6 discusses the latest advancements and research going on in scheduling; and Section 7 concludes the paper with future work.

2. BACKGROUND

The rate at which the information generated by various sources such as YouTube, facebook, twitter, LinkedIn, Google, sensors increases at a rate of 10 times for every 5 years[33]. With such huge volumes of data generated, Big data is a critical issue that requires immediate attention [32] [34]. Big data can be described with 5 characteristics: volume, variety, velocity, value and complexity according to [35]. Big data technology aims to minimize less resources and processing costs and to improve the performance in the services of business models and provides decision making support [36] [37]. Fig 1 describes the Big data management process which involves data sources, data acquisition, data processing, analysing and reporting. Data has become a crucial parameter in the growth of any organization. Data sources can be external and internal in the form of master data, transactional data, reference data, data generated by social media and machine generated data. Data is no more confined to structured data as is followed in traditional database management tools. Structured data, Semi-structured and unstructured are the forms of data generated and to be processed and analysed in Big data. Traditional database management tools has played a vital role and produced better results. Due to the tremendous increase in the data, these tools cannot process with such huge volumes generated by various sources such as web logs, sensors, YouTube, facebook ,twitter and telecommunications which has led to the evolution of Big data.

MapReduce is a framework used by Google for processing huge amounts of data in a distributed environment [45]. MapReduce [3] and its open-source implementation, Hadoop [4], have emerged as the leading computing platforms for big data management. Managing operational data for analysis includes a complex batch processing tool for extracting or capturing data, transforming it and loading into a database. The various challenges faced by BDM includes – unstructured data, scalability, accessibility, real-time processing, fault tolerance. Big Data requires efficient way to process large quantities of data within tolerable elapsed times. Data intensive computing systems such as Hadoop must be capable to provide an efficient scheduling mechanism for enhanced utilization in a shared cluster environment. A popular data processing engine for big data is Hadoop Map Reduce. The MapReduce framework became extremely popular, because organizations could deploy it on available computing components for parallel processing. Hadoop has revolutionized the world for its ability to economically store and analyse large data sets. Hadoop and its core programming model, MapReduce are the crucial for batch-oriented processing of huge amounts of data. MapReduce is useful for batch processing on terabytes or petabytes of data stored in Hadoop. Hadoop MapReduce is designed for batch processing of large volumes of data and its key advantages are simplicity, scalability, speed, recovery and minimal data motion. Hadoop works at its best for mining large volumes of historical data stored on the disk, but since the last few years, there is a tremendous growth in real-time data and thus raises need to process huge volumes of data in real-time and online processing aspects. MapReduce is not able to process recursive or iterative jobs inherently [2]. HDFS by itself is designed for high throughput data I/O rather than high performance I/O. The overhead of framework for starting a job, like copying codes and scheduling is another problem that prevents it from executing interactive jobs and real-time queries.

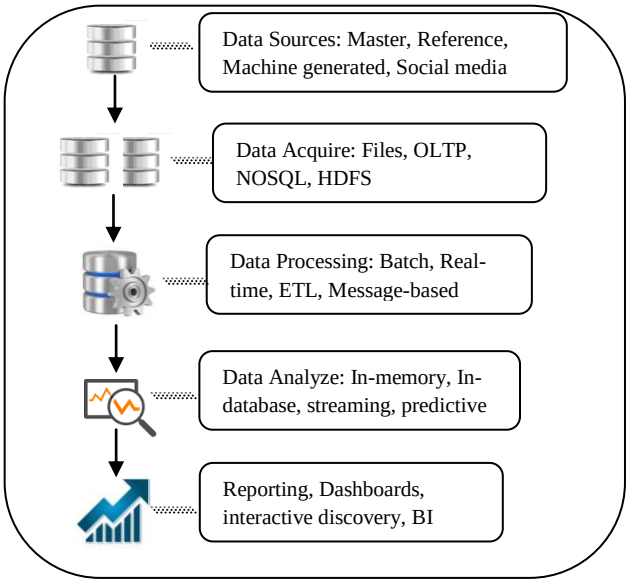

Fig 1: Big data Management Process

Big data makes it possible to handle huge volumes of data without requiring complex hardware and high cost. Several tools are available for Big data management such as Not Only SQL (NoSQL), Apache Avro, Hadoop, MemcacheDB, Google BigTable, SimpleDB and Voldemort [38]. Big data differs from the traditional data and cannot be stored in a single node. The most commonly used tools and techniques are Hadoop, MapReduce and Big Table.

Workflow management is not only about how a specific unit of work is submitted, packaged and scheduled, but is also about how it executes and how it handles failures and returns results. In Hadoop all scheduling and allocation decisions are made on a task and node slot level for both the map and reduce phases.

3. RELATED TECHNOLOGIES

Map/Reduce application mainly uses HDFS for storing data. HDFS is very large distributed file systems that assumes commodity hardware and provides high throughput and fault tolerance. Hadoop works on a distributed environment and is build to store, handle and process petabytes and Exabyte of data. Hadoop is a framework that handles large amount of data for processing. The following section describes the technologies related to Big data. Figure 2 describes the various technologies related to Big data.

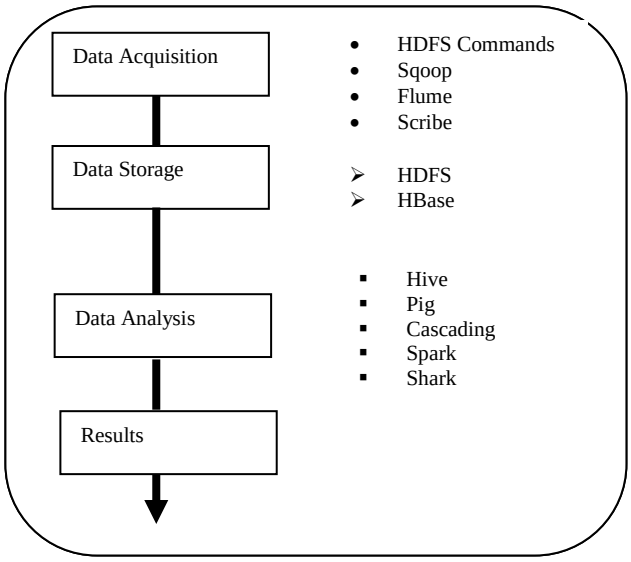

Fig 2: Technologies used at various stages of Big data management

There are six steps in Hadoop process execution: job submission, job scheduling, task assignment, task execution, process and status updates and job completion. The analysis is done with the help of two basic functionalities provided by the Hadoop framework:

MapReduce: MapReduce [41] is a framework for processing parallelizable problems across huge datasets using a large number of computers, collectively referred to as a cluster (if all nodes are on the same local network and use similar hardware) or a grid (if the nodes are shared across geographically and administratively distributed systems, and use more heterogeneous hardware). Computational processing can occur on data stored either in a file system (unstructured) or in a database (structured). MapReduce [42] takes advantage of the locality of data, data processing on or near the storage assets in order to decrease the data transmission. Figure 3 describes Hadoop MapReduce process, which involves input dat, split phase, Map phase, Intermediate data, Reduce phase and Output data.

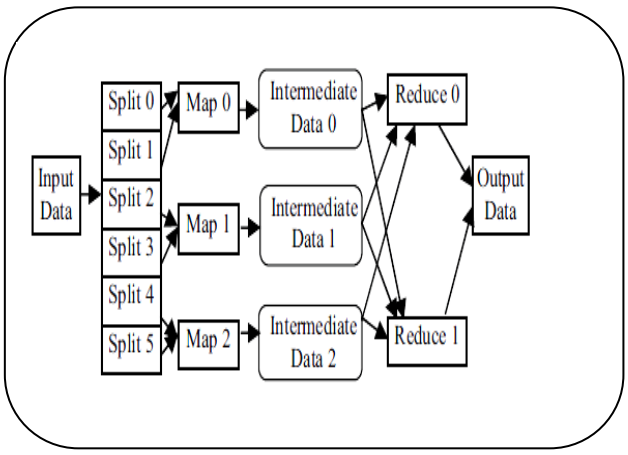

Fig 3: Hadoop MapReduce Process

HDFS: Hadoop [43] uses Hadoop distributed File System (HDFS) which is an open source implementation of the Google File System (GFS) for storing data. HDFS is a distributed file system that not only stores the data but also ensures fault tolerance through replication designed to run on commodity hardware. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant. HDFS is highly fault-tolerant and is designed to be deployed on low-cost hardware. HDFS provides high throughput access to application data and is suitable for applications that have large data sets.

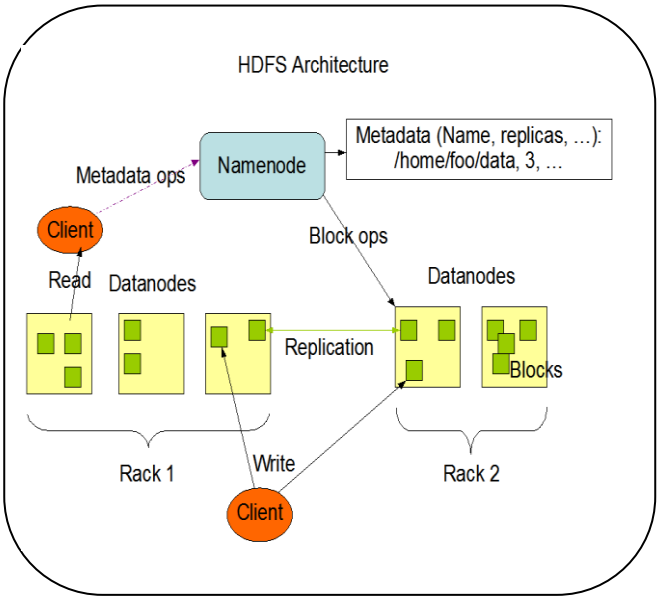

Fig 4: HDFS Architecture

Figure 4 describe HDFS has a master/slave architecture. An HDFS cluster consists of a single NameNode, a master server that manages the file system namespace and regulates access to files by clients. In addition, there are a number of DataNodes, usually one per node in the cluster, which manage storage attached to the nodes that they run on. HDFS exposes a file system namespace and allows user data to be stored in files. Internally, a file is split into one or more blocks and these blocks are stored in a set of DataNodes. The NameNode executes file system namespace operations like opening, closing, and renaming files and directories. It also determines the mapping of blocks to DataNodes. The DataNodes are responsible for serving read and write requests from the file system's clients. The DataNodes also perform block creation, deletion, and replication upon instruction from the NameNode.

4. ISSUES AND CHALLENGES

The problems of Big data can be summarized in six categories listed in Table 2. This paper identifies and presents the challenges of Big data management and analyse the solutions currently available. The related work can be summarized from six perspectives: data processing, job scheduling, power consumption, performance, resource allocation, privacy and Security.

Table 2. Problems and Challenging Issues in Big data

Category of Big data Problem	Challenging issues
Speed	real-time and offline response problems
	Statistical analysis problems

	Query and retrieval problems
	Import and export problem
Formats of data sources	Problem faced due to various source formats
	Problems due to heterogeneity in data sources
	Problems faced due to infrastructure
Volume and flexibility	Complex interactions among various resources
	Job Scheduling problems
Cost	Cost Comparison between Master node and slave node.
	Costs raised due to upgrade or modification of nodes
Storage and security	Structured and non-structured
	Data security
	Privacy security
Connectivity and data sharing	Data standards and interfaces
	Shared protocols
	Access permissions

Data processing: Big Data has two fundamental challenges: how to store huge volumes of data and how to process the data. A popular data processing engine for Big Data is Hadoop MapReduce. In today's scenario, processing with data sets with the magnitude of terabytes or even petabytes has become a reality [10] [12] [13]. Two solutions were proposed by Saeed Shahrivari and Saeed Jalili [1]. Adding real-time processing capabilities to MapReduce and providing stream processing of Big Data are the two possible solutions proposed by the author. With growing data, Hadoop enables to horizontally scale clusters by adding commodity nodes and thus keep with query workloads and make it faster to enable execution of jobs in very less time. In-memory MapReduce can be up hundreds of times faster than running it on a disk as is traditional for systems like Hadoop and which is based on using distributed memory system to store and process Big Data in real-time. Storm from Twitter and S4 from Yahoo [3] are two predominant stream processing frameworks. The main advantages of these two are that they can run on a Java Virtual Machine.

Job scheduling: Job scheduling is one of the core technologies of Hadoop MapReduce and its main function is to control the order of job execution and assign user's job to run up on the resources. The default scheduling algorithm is based on FIFO where jobs are executed in the order of their submission. The limitation of the FIFO Scheduler is the balance of resource allocation between long jobs and short jobs is not taken into account. Facebook and Yahoo contributed significant work in developing schedulers: Fair Scheduler [6] and Capacity Scheduler [7]. The fair scheduler can limit the number of concurrently running jobs from each user and from each pool. Running job limits are implemented by marking jobs as not runnable if there are too many jobs submitted by the same user or pool. If there is a single job running, the job uses the entire cluster. Capacity Scheduler is designed to run Hadoop applications as a shared, multi-tenant cluster in operator-friendly fashion while maximizing the throughput and the utilization of the cluster. The capacity scheduler supports the features of hierarchical queues, capacity guarantees, elasticity and operability. In Delay scheduling [8], it considers two approaches in reassigning resources: killing tasks from existing jobs to allocate new jobs and waiting for tasks to finish assigning slots to new jobs.

Power consumption: The increase in power consumption of data-centres is a challenging issue in Big Data Management. MapReduce clusters constitute a major part of data-centres for Big Data processing applications. The factors that influence the energy consumption are sheer size, high fault-tolerant nature and low utilization levels. Efficient use of energy in MapReduce clusters can contribute significantly towards energy efficiency of MapReduce framework. Willis Lang and Jignesh M.Patel [9] aim to improve the energy efficiency of the MapReduce clusters to exploit low utilization periods that turn off nodes to reduce the energy consumption when the overall system utilization drops. The impact of the workload characteristics, hardware characteristics and performance targets are considered to bring out the interactions between these factors and cluster energy consumption.

Performance: There is a lot of research is going on in order to optimize the Hadoop jobs performance and the efficiency of Hadoop clusters in different aspects [11] [14] [15]. Dawei Jiang et al conducts performance study on MapReduce with aspects such as impact of architectural design of MapReduce, storage-independent design and identifies five factors that affect the performance of MapReduce: Indexing, I/O mode, parsing of data, grouping schemes and block-level scheduling. In [18], an extensive experiment was conducted on job configuration parameters that affect the performance of Hadoop MapReduce jobs. Hadoop. Jiong Xie et al [19] addresses the problem of data locality and proposes data placement scheme adaptively that is capable of improving data-processing performance by a data placement scheme that distributes and stores data across multiple heterogeneous nodes based on their computing capacities. The initial data placement algorithm proposed by the authors begins by dividing a large input file into several even-sized fragments and assigns fragments to nodes in a cluster. The data redistribution procedure is described as collection of disk space utilization of a cluster by data distribution server which creates two node lists: number of local fragments in each node.

Resource allocation: The resource allocation in Hadoop Mapreduce is done at level of fixed-size resource splits of the nodes called slots. This mechanism has disadvantages of under and over utilization of resources. In [20], the authors identify the efficient resource management across various data centres and clouds running large distributed data processing frameworks are a crucial for enhancing the performance of MapReduce applications. Jorda Polo et al [21] propose a resource-aware scheduling technique, Resource-aware Adaptive Scheduler (RAS) for MapReduce multi-job workloads which is designed to improve the resource utilization across nodes by obtaining job's completion time. The jobs are dynamically adjusted on each node to maximize the resource utilization. RAS is a novel resource resource management and job scheduling scheme for Hadoop MapReduce.

Privacy and Security: Security has become the most crucial phase in any organization. In the era of Big Data, huge volumes of data are collecting, analyzing and reporting the decisions based on the analysis from various sources, there must be strong secure mechanism to protect the privacy of the data. A study released by Symantec and Ponemon Institute found that the average organizational cost of one security breach in the United State is 5.4 million dollars [22]. In India, Aadhaar Project is the biggest project in the human history to collect the details of the citizens and allocating a unique identity to each citizen. Security and privacy are the two mandatory requirements of this kind of Big Data in order to protect the rights of the citizens from the attackers. Travis Mayberry et al [23] overlooked the problem with outsourcing data to the cloud is the privacy of access patterns. The authors propose Private Information Retrieval MapReduce (PIRMAP) which allows a user to retrieve data from a database while hiding the user's access pattern. Large computational resources that are available in cloud settings are considered in this solution that can run efficiently on MapReduce and scale to a large number of nodes.

5. JOB SCHEDULING

Till 2008, Hadoop supported a single scheduler only. Currently, Hadoop configuration is based on cluster hardware information and the number of nodes can greatly improve the performance of Hadoop clusters. Hadoop implements the ability for pluggable schedulers that assign resources to jobs. Users can design their own dispatchers according to the actual application requirements [44].

Our research focuses on Hadoop MapReduce Job Scheduling along with challenges in MapReduce. Hadoop MapReduce is a programming model for processing and generating large datasets [17]. Hadoop is a multi-tasking system that can process multiple data sets for multiple jobs for multiple users at the same time. This capability of multi-tasking makes Hadoop an opportunity to optimally map jobs to resources. Hadoop operates in a batch mode, where jobs were submitted to a queue an infrastructure to execute them in the order of receipt. MapReduce framework is scheduled by JobTracker and TaskTracker [24]. The relationship of tasks allocation is shown in Fig. 2. JobTracker is the only master control, which can run on any computer in the cluster for scheduling and managing other TaskTrackers, allocating Map task and Reduce task to free TaskTrackers for parallel running and monitoring the condition of the tasks. There can be more than one TaskTracker. TaskTracker is in charge of the implementation of the tasks [23]. It must run on DataNode, which means that DataNode is not only a data storage node, but also a computing node. If a TaskTracker's task fails, JobTracker will allocate the task to one of other free TaskTrackers, and rerunning [25]. When a job is submitted to the MapReduce framework, MapReduce will divide it into several Map tasks and assign them to different nodes for running. Every Map task only deals with a part of the input data. After Map task processing, the results, those intermediate state key-value pairs, will be sent to the Reduce function. Reduce function will merge the pairs based on a specific key, then generate and output the value-keys that client requires.

Various scheduler algorithms used in Hadoop MapReduce namely FIFO Scheduler, Fair Scheduler, Capacity Scheduler are summarized.

FIFO Scheduling: FIFO scheduler was integrated within Job tracker. The Each job uses whole cluster and so jobs must wait for their turn. FIFO: First In First Out, based on which ever job comes first in the queue, it is processed and then pulls the next job in the queue. This follows priority and submission at the same time, which leads to starvation of jobs in the presence of a long running job. The other disadvantages of this scheduler are poor utilization, costly data replications, data locality and no priority or size of the job.

Fair Scheduling: Fair Scheduling is a method of assigning resources based on the resource pool, which groups jobs into pools and performs fair sharing between these pools. The Fair scheduler was designed to meet the following four main objectives: run small jobs quickly even if they are sharing a cluster with large jobs, provide guaranteed service levels to production jobs, easy to administer and configure, support reconfiguration at run-time.

Capacity Scheduling: Capacity scheduler is a pluggable MapReduce scheduling algorithm for Hadoop which provides a way to share large clusters. In Capacity scheduling, the job queue is divided into several jobs and each job queue still runs by the way of FIFO. This provides greater control to provide a minimum capacity guarantee and share excess capacity among users. In Capacity scheduling, instead of pools, several queues are created, each with a configurable number of map and reduce slots. Each queue enforces a limit on the percentage of resources that a given user can access of multiple users are accessing the queue at the same time. The user-limits are dynamic and the actual limits depend upon the active users and their demand. The capacity scheduler also supports high resource applications such that MapReduce job can obtain multiple slots for every map or reduce task.

5.1 Analysis of the Scheduling Algorithms

FIFO Scheduling: Hadoop MapReduce follows FIFO as the default scheduling algorithm. The advantages of FIFO includes easy and simple for execution, less workload on the job server. The drawbacks of FIFO are that it ignores the different needs by different operations. When there is a need for analysing huge volumes of data which occupies computing resources for a long time, then subsequent interactive operations may lead to long response time and affect the hit-time response. FIFO scheduling is best suited for batch systems, which follows non-preemptive method. It implements one queue which holds the tasks in the order they come in. The order of the task arrival is very important for average turnaround time.

Fair Scheduling: The Fair Scheduler can limit the number of concurrent running jobs per user and per pool. It can also limit the number of concurrent running tasks per pool. If a pool does not get its minimum share for long time, Fair scheduling pre-empts the most recently started task of an over allocated job from some other pool. This ensures that a long running job do not block the execution of some production jobs. The priorities are used to assign job weights and resources are allocated as per the normalized fractions to the jobs. Fair scheduling works well for both the cases of small and large clusters. Fair Scheduler is that it does not consider the job weight of each node.

Capacity Scheduling: The objective of the Capacity scheduling is to maximize the resource utilization and throughput in multi-tenant cluster environment. Instead of pools as in Fair Scheduler, Capacity scheduler uses queues. Each queue is assigned to an organization and resources are divided among these queues. In order to control access to the queues, security mechanisms are built so that each organization can access only its queue and it cannot interrupt

with other organization's queues or jobs. Capacity scheduling offers minimum capacity guarantee by having limits on running tasks and jobs from a single queue. It allows resource re-allocation to queues using their full capacity inorder to maximize resource utilization. When jobs arrive in that queue, running jobs are completed and resources are pushed back to the original queue. This allows priority based scheduling of jobs in an organization queue.

6. IMPROVEMENTS IN JOB SCHEDULING

Delay Scheduling [26] is an outcome of strict implementation of fair sharing compromising locality. To resolve this problem of locality, Delay scheduling algorithm was proposed, in which a job waits for a limited amount of time for a scheduling opportunity on a node that has data for it. The main goal of Delay Scheduling is to statistically multiplex clusters while maintaining minimal impact on fairness and achieving high data locality.

Delay scheduling algorithm temporarily relaxes fairness to improve locality by asking jobs to wait for a scheduling opportunity on a node with local data. Two locality problems were identified from fair scheduler are: head-of-line scheduling and sticky notes. The first locality problem occurs in small jobs. Whenever a job reaches the head of the sorted list for scheduling, one of its tasks is launched on the next slot that becomes free irrespective of which node the slot is on. The pseudocode for this algorithm is shown below:

Algorithm: Delay Scheduling

```
When a heartbeat is received from a node n:
 if n has a free slot then
   sort jobs in increasing order of number of running tasks
   for j in jobs do
    if j has unlaunched task t with data on n then
     launch t on n
   else if j has unlaunched task t then
      launch t on n
    end if
   end for
 end if
```

In Delay Scheduling, it identifies the impact job response times significantly if at least one of the following conditions holds:

- When there are many jobs running, each job's fractional share of the cluster,
- Jobs with a small number of tasks,
- Jobs where jobs (J) is greater than average task length (T) incurs with little overhead.

The problem of enforcing strict queuing order forces a job with no local data to be scheduled is addressed by a simple technique called Delay Scheduling. When a node requests a task, if the head-of-line job cannot launch a local task, it skips and looks for the next subsequent jobs. If a job is skipped for long enough time, it starts allowing it to launch non-local tasks in order to avoid starvation. There should be a mechanism that is to be involved when once a job has been skipped for some number of times; it must launch arbitrarily many non-local tasks without resetting its count.

MapReduce workloads may be heterogeneous in terms of their data size and their resource requirements [10]. Bogdan Ghit et.al [11], proposes FAWKES in which balance resources across the MapReduce clusters which is capable to resize them by growing and shrinking the number of resources allocated to them at runtime. The most important requirement for FAWKES is to provide reliable data management so that when nodes are removed from an MapReduce cluster and the number of replicas is small, no data are lost. To enable fast reconfigurations, the removed nodes of the MapReduce cluster should store relatively small amounts of data.

Genetic algorithm [27] provides the solution in selecting the operation sequence, to avoid the time overhead between frequent switching of shorter jobs, adapting long jobs and setting up a task queue length limit to meet the demand. In genetic algorithm, it uses fitness function for the average time of finishing a job. The following are the steps of the algorithm:

Algorithm: Genetic Scheduling

```
M jobs are waiting for running in queue
   m jobs enter into segmentation into s tasks
   if s > L && m!=1 then
    m=m-1
     goto first step
     else
      Select task order based on GA
      Put s tasks into task queue
      Running ends
      Check for any other jobs in queue
       if yes, goto first step
        else Stop.
    end if
end if
```

Longest Approximate Time to End (LATE) [28] algorithm improves the execution time by identifying real slow tasks. The algorithm computes the remaining time of all tasks and selects a set of tasks with longer remaining time when compared to all the nodes and considers them as real slow tasks. LATE algorithm works as follows:

Algorithm: Genetic Scheduling

```
If a task slot becomes available and there are less than
speculative cap speculative tasks running:
  Ignore the request if the node's total progress is below
    SlowNodeThreshold
  Rank currently running, non-speculative executed tasks by
    estimated time left
  Launch a copy of the highest-ranked task with progress rate
below SlowTaskThreshold
```

LATE is based on three principles: prioritizing tasks to speculate, choosing fast nodes to run, and capping speculative tasks to prevent thrashing. The advantages of LATE algorithm includes: robust to node heteroginity, prioritizes among slow tasks, focuses on estimated time left rather than progress rate. The drawback of LATE is that if some commodity hardware node is running behind its peers, instead of trying to finding out the reasons on its behaviour, it marks as a straggler. The complications include temporary defect or permanent crippling on the tasks during the entire duration of computation.

Jiang, Ooi, Shi and Wu [16] have explained the importance and performance of MapReduce. Five design factors were identified that affect Hadoop performance: 1) grouping schemes, 2) I/O modes, 3) data parsing, 4) indexing, and 5) block-level scheduling. The overall performance can be improved by tuning these factors.

Babu [46] highlights the combination of MapReduce frameworks and cloud computing and measured a real system that the presence of too many job configuration parameters in Hadoop is unwieldy. The reactive and competitive approaches reduce or eliminate the need for cost models. An automated tool is introduced to optimize parameter values by setting of job configuration parameters for MapReduce programs.

Herodotou et al. [47] introduced Starfish, which profiles and optimizes MapReduce programs based on cost. Starfish aims to relieve users from having to fine tune job configuration parameters for different cluster settings and input datasets. All these profiling tools require a real system on which to test potential workloads and are tuned to the job scheduler and not the task scheduler.

Two short comings of where data and computational resources are shared and accessed were explained: tight coupling of specific programming mode with the resource management infrastructure, forcing developers to comment on the performance of MapReduce programming model and centralized handling of jobs [48]. Yet Another Resource Negotiator (YARN), the new architecture were introduced which is capable of decoupling the programming model from the resource management infrastructure. Several discrete event simulators for MapReduce application workloads have been developed.

Each provides different levels of support for integrating new task schedulers and different details in the computation and communication models. SimMR [49], MRSim [50] and SimMapReduce [51] are designed to evaluate different schedulers/provisioning strategies. SimMR focuses on JobTracker decisions and task/slot allocations among different jobs. MRSim measures scalability easily and captures the effects of different configurations of Hadoop setup on job completion times and hardware utilization. SimMapReduce allows researchers to evaluate different scheduling algorithms and resource allocation policies.

Hsim [52] simulates the dynamic behaviours of Hadoop environments and models a large number of Hadoop parameters such as node parameters (related to processors, memory, hard disk, network interface, map and reduce instances) and cluster parameters, (number of nodes, node configurations, network routers, job queues and schedulers), and Hadoop system parameters.

MRPerf [20, 21] analyzes application performance on a given Hadoop setup, enabling the evaluation of design decisions for fine-tuning and creating Hadoop clusters. MRPerf was made open source to be used by the research community to enable exploration of design issues, validation of new algorithms and optimization in MapReduce. The need for production level traces by some simulators makes them inappropriate for general research, since often the traces are proprietary information and not easily available to the academic community.

Only recently have MapReduce simulators supported schedulers other than FIFO, Mumak being the first. A new simulation environment, SLS5 has been recently introduced by Yahoo! for the YARN framework includes support for many components within Hadoop. SLS provides detailed execution traces as well as resource usage metrics. It even provides analysis of low-level scheduler operations, measuring their overhead and assessing scalability.

The environment is designed in a modular fashion to incorporate new scheduler development. There are many parameters in the simulator itself. The goals and approach of our work and SLS are similar, but SLS was not a mature tool at the beginning of our investigation. Optimizing cluster utilization and reducing makespan has been studied using job scheduler adjustments.

7. CONCLUSION AND FUTURE WORK

This paper makes an attempt to present background of Big data management, technologies used at various stages of BDM and discussed various issues and challenges faced in processing huge volumes of data. To be able to process large-scale datasets, the fundamental design of the standard Hadoop places more emphasis on high throughput of data than on job execution performance. This causes performance limitation when Hadoop MapReduce is used to execute short jobs that requires quick responses. In order to speed up the execution of short jobs optimization methods are required to improve the execution performance of MapReduce jobs. Three important scheduling issues in Hadoop MapReduce are identified: data locality, synchronization and fairness. This paper makes a comprehensive survey on Big data, Big data management and job scheduling algorithms and highlights the importance scheduling in Hadoop MapReduce. Various parameters that affect the performance of Hadoop MapReduce are analyzed with the perspective of job scheduling. Our future work includes developing a new job scheduling algorithm considering of all the parameters describe in this paper which can yield better performance.

8. REFERENCES

[1] Saeed Shahrivari and Saeed Jalili, "Beyond Batch Processing: Towards Real-Time and Streaming Big Data," in Computers 2014, pp. 117 – 129, doi: 10.3390/computers3040117.

[2] J. F.N Afrati, V. Borkar, M. Carey, N. Polyzotis, and J. D. Ullman, "Map-reduce extensions and recursive queries," in 14th International Conference on Extending Database Technology, 2011, pp. 1–8.

[3] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," in *6*th USENIX Symp. Oper. Syst. Des. Implementation, 2004, pp. 137-150.

[4] Hadoop. (2014) [Online]. Available: http://hadoop.apache.org/.

[5] L. Neumeyer, B. Robbins, A. Nair, and A. Kesari, "S4: Distributed Stream Computing Platform," in Proceedings of IEEE International Conference on Data Mining Workshops (ICDMW), 2010, pp. 170–177.

[6] Yang XIA, Lei WANG, Qiang ZHAO and Gongxuan ZHANG, "Research on Job Scheduling Algorithm in Hadoop," in Journal of Computational Information Systems 7:16, pp. 5769 – 5775, December 2011.

[7] Capacity Scheduler for Hadoop [EB/OL]. *http://Hadoop. apache.org/common/docs/current/Capacity_scheduler. .html, 2010-03-22*

[8] Matei Zaharia, Dhruba Borthakur and Joydeep Sen Sarma, "Delay Scheduling: A Simple Technique for Achieving Locality and Fairness in Cluster Scheduling," in EuroSys'10 ACM International Conference, Apr 13 – 16, 2010.

[9] Willis Lang and Jignesh M. Patel, "Energy Management for MapReduce Clusters," in International Conference on Very Large DataBases, Proceedings of the VLDB Endowment, Vol. 3 No.1, September 2010.

[10] A. Thusoo , "Hive: A Petabyte Scale Data Warehouse Using Hadoop," in Proc. ICDE, pp. 996-1005, 2010.

[11] M. J. Fischer, X. Su, and Y. Yin, "Assigning tasks for efficiency, hadoop: extended abstract", in Proc., SPAA, 2010, pp. 30-39 .

[12] A. Thusoo, "Data Warehousing and Analytics Infrastructure at Facebook", in Proc., ICDE, pp. 1013-1020, 2010.

[13] D. Logothetis, Statefull Bulk Processing for Incremental Analytics," in Proc.,SOCC, pp.51-62, 2010.

[14] A. Verma, L. Cherkasova, and R. H. Campbell, "Aria: automatic resource, inference and allocation for mapreduce environments," in Proc., 8th ACM international conference on Autonomic computing, ICAC'11, USA: ACM, 2011, pp. 235-244. [Online]. Available: http://doi.acm.org/10.1145/1998582.1998637.

[15] J. Polo, D. Carrera, Y. Becerra, M. Steinder and I. Whalley, "Performance-driven task co-scheduling for mapreduce environments," in Proc., NOMS, 2010, pp. 373-380.

[16] Dawei Jiang, Beng Chin Ooi, Lei Shi and Sai Wu, "The Performance of MapReduce: An In-depth Study," in Proc., VLDB Endowment, Vol. 3, No. 1, 2010, pp.472-483.

[17] J. Dean and S. Ghemawat, "MapReduce: A flexible data processing tool," in Proc ACM, 53(1), 2010, pp. 72-77.

[18] S. Babu, "Towards automatic optimization of mapreduce programs," in proc., SoCC, ACM, 2010, pp. 137-142.

[19] Jiong Xie, Shu Yin, Xiaojun Ruan, Zhiyang Ding, Yun Tian, James Majors, Adam Manzanares and Xiao Qin, " Improving MapReduce Performance through Data Placement in Heterogeneous Hadoop Clusters," in Proc., 19th International Heterogeneity in Computing Workshop, Atlanta, Georgia,2010.

[20] Bikash Sharma, Ramya Prabhakar, Seung-Hwan Lim, Mahmut T. Kandemir and Chita R. Das, "MROrchestrator: A Fine-Grained Resource Orchestration Framework for Hadoop MaReduce," Technical Report, CSE-12-001, January 2012.

[21] Jorda Polo, Claris Castillo, David Carrera, Yolanda Becerra, Ian Whalley, Malgorzata Steinder, Jordi Torres, Eduard Ayguade, "Resource-Aware Adaptive Scheduling for MapReduce Clusters," in LNCS Vol. 7049, Springer, 2011, pp 187-207.

[22] Ponemon Institue, "2013 Cost of Data Breach Study: Global Analysis," May 2013.

[23] J. Dean, and S. Ghemawat, "MapReduce: simplified data processing on large clusters," ACM, vol. 51, no. 1, pp. 107-113, 2008.

[24] T. White, Hadoop: The Definitive Guide: O'Reilly Media, 2009.

[25] Changqing Ji, Yu Li, Wenming Qiu, Uchechukwu Awada, Keqiu Li," Big Data Processing in Cloud Computing Environments", 2012 International Symposium on Pervasive Systems, Algorithms and Networks.

[26] Matei Zaharia, Dhruba Borthakur and Joydeep Sen Sarma, "Delay Scheduling: A Simple Technique for Achieving Locality and Fairness in Cluster Scheduling", in proc., EuroSys'10, ACM, 2010.

[27] Xueying Jiang, Zhongyao Li and Yang Yang, "Implementation of a Hadoop platform scheduling algorithm based on a genetic algorithm," WIT Transactions on Information and Communication Technologies, Vol. 55 , pp. 595 – 605, 2013.

[28] M. Zaharia, A. Konwinski, A. D. Joseph, R. Katz, and I. Stoica, "Improving mapreduce performance in heterogeneous Environment," in proc., IEEE 10th Internation Conference on CIT, pp. 2736-2743, 2010.

[29] H. Moed, "The Evolution of Big Data as a Research and Scientific Topic: Overview of the Literature," 2012, Research Trends, http://www.researchtrends.com.

[30] S. S. Kaisler, F. Armour, J. A. Espinosa, andW.Money, "Big data: issues and challenges moving forward," in Proceedings of the IEEE 46th Annual Hawaii International Conference on System Sciences (HICSS '13), pp. 995–1004, January 2013.

[31] R. Cumbley and P. Church, "Is "Big Data" creepy?," Computer Law and Security Review, vol. 29, no. 5, pp. 601–609, 2013.

[32] J. M. Wing, "Computational thinking and thinking about computing," Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, vol. 366, no. 1881, pp. 3717–3725, 2008.

[33] S. Hendrickson, "Getting Started with Hadoop with Amazon's Elastic MapReduce," EMR, 2010.

[34] J.Mervis, "Agencies rally to tackle big data," Science, vol. 336, no. 6077, p. 22, 2012.

[35] S. Sagiroglu and D. Sinanc, "Big data: a review," in Proceedings of the International Conference on Collaboration Technologies and Systems (CTS '13), pp. 42–47, IEEE, San Diego, Calif, USA,May 2013.

[36] J. Manyika, C. Michael, B. Brown et al., "Big data: The next frontier for innovation, competition, and productivity," Tech. Rep., Mc Kinsey, May 2011.

[37] J. Manyika, M. Chui, B. Brown et al., "Big data: the next frontier for innovation, competition, and productivity," McKinsey Global Institute, 2011.

[38] M. Chen, S. Mao, and Y. Liu, "Big data: a survey," Mobile Networks and Applications, vol. 19, no. 2, pp. 171–209, 2014.

[39] Zikopoulos PC, Eaton C, DeRoos D, Deutsch T, Lapis G. "Understanding big data," New York et al: McGraw-Hill, 2012.

[40] Bell G, Hey T, Szalay A. "Beyond the data deluge," Science, 2009, 323(5919): 1297-1298.

[41] Narasimhaiah Gorla and Kang Zhang, "Deriving Program Physical Structures using Bond Energy algorithm," in Proceeding 6th Asia Pacific Software Engineering Conference,pp-359,1999.

[42] Dong Yuan, Yun Yang, Xiao Liu, Jinjun Chen, "A Data Placement Strategy in Scientific cloud workflows," pp-1200-1214, 2010.

[43] Xie Jiong, Yin Shu, Ruan Xiaojun, Ding Zhiyang, Tian Yun, "Improving Mapreduce performance through data placements in heterogeneous hadoop cluster," 2010.

[44] Huang Lu, Hu Ting-ting and Chen Hai-shan, "Research on Hadoop Cloud Computing Model and its Applications," 2012 Third International Conference on Networking and Distributed Computing.

[45] D. Jiang, B. C. Ooi, L. Shi, and S. Wu "The performance of MapReduce: an in-depth study," VLDB Endowment, 3(1-2):472–483, Sept. 2010.

[46] S. Babu "Towards automatic optimization of MapReduce programs," in IEEE SoCC , pages 137– 142, Indianapolis, June 2010.

[47] H. Herodotou, H. Lim, G. Luo, N. Borisov, L. Dong, F. B. Cetin, and S. Babu, "Starfish: A Self- tuning System for Big Data Analytics," in CIDR, pages 261–272, Asilomar, CA, Jan. 2011.

[48] Vavilapalli, A. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, B. Saha, C. Curino, O. O'Malley, S. Radia, B. Reed, and E. Baldeschwieler, "Apache Hadoop YARN: Yet Another Resource Negotiator," in IEEE SOCC, pages 5:1–5:16, Santa Clara, CA, Oct. 2013.

[49] A. Verma, L. Cherkasova, and R. H. Campbell, "Play it Again, SimMR!," in IEEE CLUSTER, pages 253–261, Austin, TX, Sept. 2011.

[50] S. Hammoud, M. Li, Y. Liu, N. K. Alham, and Z. Liu, "MRSim: A discrete event based MapReduce simulator," in Fuzzy Systems and Knowledge Discovery, pages 2993–2997, Yantai, China, Aug. 2010.

[51] F. Teng, L. Yu, and F. Magoulaas, "SimMapReduce: A simulator for modeling MapReduce frame-work," in FTRA Mobile and Ubiquitous Engineering, pages 277–282, Crete, Greece, June 2011.

[52] Liu, M. Li, N. K. Alham, and S. Hammoud, "HSim: A MapReduce Simulator in Enabling Cloud Computing," in Future Gener. Comput. Syst. 29(1):300–308, Jan. 2013.